08｜故障复盘：黄金三问与判定三原则

2020-04-03 赵成

SRE实战手册 进入课程 >

讲述：赵成
时长 11:07　大小 10.19M

你好，我是赵成，欢迎回来。

前两讲，我们聚焦在 MTTR 阶段，我跟你分享了从故障发现到故障处理的一些经验。但是，即便我们身经百战，做足了准备，故障的发生依然是很难避免的。这时候，我们也没必要太沮丧，SRE 中有一条很重要的原则可以帮到我们，那就是“从故障中学习和提升”，也就是故障复盘。

那么，今天我会专门用一节课，来和你分享我在故障复盘过程总结的经验。

故障复盘的黄金三问

提起故障复盘，我自己的团队也是踩了很多坑，说出来都是血泪史。

最开始，我们坚信既然要复盘，那就一定要追根溯源，找到根因，最好是一次性解决所有问题，一次性把事情做对肯定是最高效的呀。

但是，在执行的过程中，我们发现，对于根因的理解和定义，每个人或每个角色都不一样。而且，一旦设定为找根因，那就只能有一个，还特别容易根据找到的根因来定责，导致把原本的寻求根因是什么，转变为“责任是谁”的问题。本来是想通过复盘来引导改进的，但是很容易画风一变，开始推诿扯皮，故障复盘会就开成了批斗会，每个参与的人都要承受很大的心理压力。

我这么说，不知道你是不是有同感。接下来我给你讲两个特别常见的情况，你也感受下。

比如，服务器故障导致上面业务也发生问题了，从业务开发同学的角度来看，这肯定是因为服务器不稳定造成的呀，根因就是服务器故障啊！但是从系统维护同学的角度来看，硬件故障属于不可控事件，所以这种情况下，根因应该是业务应用层没有做到高可用。你看，不同的角度，不同的分析判断。就这个情况来说，你觉得根因是什么？

再比如，网络瞬时抖动了一下，导致很多业务请求失败，严重的还导致了请求阻塞，服务不可用。那这种情况下根因是什么？是网络不好？还是业务应用没有做好容错和重试这种高可用机制？

上面这两种情况还算简单的。如果我们已经用到了公有云或私有云，基础设施都已经完全是第三方提供的时候，问题就不单单是内部的责任之争了，还会涉及甲乙双方之间的定责，必定会出现更多大的利益争执。这样的争执多了，一地鸡毛，但是对改进没有任何帮助。

后来我们的故障复盘做得越来越多，发现在真实的场景下，造成故障发生的因素确实会有很多；而且故障处理时间过长，迟迟无法恢复的原因也会有很多，如果再出现了衍生故障，可能又会有各种其他的原因。

经历过这样很痛苦的阶段后，我们总结出一条很重要的经验和观点，甚至是颠覆你对故障根因认知的观点：**故障根因不止一个**，与其争论根因是什么，不如一起看看引起故障的原因都有哪些，是不是都有改进的空间。

思路转变后，我们会将所有导致故障和衍生故障发生、业务恢复过程中耗时较长、以及做出错误决策的原因和环节都提炼出来，把这些都算是故障原因，然后针对这些问题和环节制定改进措施。

那这些原因和环节该怎么找呢？我们在做故障复盘的时候，首先会结合 Timeline 来做，也就是按照 MTTI、MTTK、MTTF 和 MTTV 先做一个分类；然后针对耗时长的环节，反复讨论改进措施是什么；最后定好责任人和时间点，后续持续跟进执行状况。

如果把这些经验再提炼一下，那就是我们总结出来的故障复盘黄金三问：

第一问：故障原因有哪些？

第二问：我们做什么，怎么做才能确保下次不会再出现类似故障？

第三问：当时如果我们做了什么，可以用更短的时间恢复业务？

具体开复盘会的时候，就是紧扣着这三问来进行的。除此之外不允许出现相互指责和埋怨的情况，如果出现，会议主持者要及时控制并打断。会议主持者这个角色，一般情况下，跟我们上一讲提到的 CL（Communication Lead），也就是“沟通引导”是一个角色，在我们公司内部叫技术支持。

故障判定的三原则

通过黄金三问，我们找到了故障发生的原因，也明确了做什么可以优化，那接下来就是落地了。要落地，就要明确到底应该**由谁来承担主要的改进职责**。注意，这里我没有用谁应该承担主要责任，而是承担主要的改进职责，也就是由谁来执行改进措施。

具体怎么来做呢？我们制定了一些故障判定原则，最重要的就是下面这三条。

1. 健壮性原则。

这个原则是说每个部件自身要具备一定的自愈能力，比如主备、集群、限流、降级和重试等等。例如，在 B 依赖 A 的状态下，被依赖方 A 出现问题，但是能够快速恢复，而依赖方 B 无法快速恢复，导致故障蔓延。这时，承担主要责任的是依赖方 B，而不是被依赖方 A。

我们前面介绍的服务器和网络类故障，其实就适用这个原则。如交换机故障发生主备切换导致的网络瞬时或短暂抖动，从网络设备这个组件来说，自身是有自愈能力的，而且在极短的时间内就可以恢复。如果应用因为抖动而失败、无法自愈，这种情况，业务应用就不能以服务器或网络故障为理由，不承担改进职责。

再比如，我们之前讲的强弱依赖问题。原则上，核心应用对非核心应用的依赖必须要有降级和限流手段。如果因为非核心应用的故障或者瞬时高并发访问，导致核心应用故障，这种情况下，主要的改进责任在核心应用，非核心应用只需要配合改造。

2. 第三方默认无责。

这一条是上一条的延伸，如果使用到了第三方的服务，如公有云的各类服务，包括 IaaS、PaaS、CDN 以及视频等等，我们的原则就是默认第三方无责。

这种涉及第三方服务的情况，在判定改进责任时会分为两部分，对内谁的服务受影响谁改进，并对外推进第三方改进，但是一定要按照我们之前讲的，稳定性一定要做到相对自我可控，而不是完全依赖外部。

比如，一个应用使用了 CDN 服务，如果一家 CDN 厂商服务出现问题，要做到随时可以切换到另外一到两家，这时就不能以某家 CDN 服务故障为由不做改进。如果用到了云存储，如 S3、OSS 或 COS 这样的服务，一定要保证存储有主备，当一个区域有问题时，也要做到可切换，甚至容忍一定的业务有损，但是必须保障全量没有大问题。

如果再提升下高可用视角，就要考虑做到双活或多活，单个 Zone 甚至单个 Region 出现问题时，也能做到切换。

对内，默认第三方无责，稳定性责任一定是内部角色承担，这样做有时看起来虽然不太合理，但这样做的目的，就是让内部意识到稳定性和高可用一定是我们自己的责任，决不能依赖任何一个第三方。这就相当于一个国家的军事国防，可以跟外部形成统一战线，可以做联合演习共同防御，但是绝不可能完完全全交给另外一个国家去建设和控制。

同时，这也是防止业务上云后，内部将大量的问题丢给外部云厂商去承担，甚至是让云厂商“背锅”，久而久之，内部员工对于稳定性的责任心就丢失掉了。

3. 分段判定原则。

这个原则主要应用在情况比较复杂的场景。当发生衍生故障，或者故障蔓延的原因与触发原因不同时，我们会将一次故障分段判断。比如我们在上一讲提到的大促故障案例，前半段是由于模型评估不准，没有故障隔离手段造成的，但是后半段，就是因为 DBA 的误操作，以及流程机制不完善造成的。

这样分段判定会让故障问题更聚焦，改进措施也会更有针对性。

讲到这里，故障判定的一些关键原则就讲完了。 做个小结，这些原则的根本出发点就是希望你摒弃 **“根因只有一个”** 这个的观点，以更开放的心态去寻找不足，而且要从自身找不足，目的就是找到更多可以改进的地方，不断从“故障中学习和改进”。

总结

今天的内容就讲完了，我们再来复习下。在做故障复盘时，我的经验和建议是：不要纠结于故障根因到底是哪个，而是把更多的注意力放在做哪些事情，可以提升故障处理的效率，缩短业务故障时长。

为了达到这个目的，我们定义了故障复盘的黄金三问，同时还设定了三个判定原则，明确改进措施的主要职责应该由谁来承担。希望通过这些具体的问题和原则，能帮助你做好故障复盘。

关于故障，我再说几句掏心窝子的话：“故障是系统运行的常态，正常才是特殊状态”。所以，你无论作为什么角色，一定要以一颗平常心来对待故障，能做到这个程度不容易，这就需要我们更加辩证地看待故障，我们一定要做到鼓励改进，而不是处罚错误。

思考题

最后，给你留一个思考题。

你可能还听说过 5W 分析法，就是针对故障复盘至少问 5 个为什么，通常就可以找到比较深层次的原因，或者是根因。你有没有用过 5W 的方法呢？你觉得 5W 方法是不是一个好的故障复盘分析手段呢？

欢迎你留言分享自己的思考，也可以把今天的内容分享给你的朋友，和他一起学习。

我是赵成，我们下节课见！

© 版权归极客邦科技所有，未经许可不得传播售卖。 页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇　07｜故障处理：一切以恢复业务为最高优先级

下一篇　09｜案例：互联网典型的SRE组织架构是怎样的?

精选留言 (8)

写留言

leslie
2020-04-03

其实这里提及到一个比较重要的点：备用措施。老师今天的课程又再次提及了原来课程推荐的《SRE Google 运维解密》中的经典名言“故障是常态，正常是特殊态”。
要想避免故障最关键的其实就是“是否有备用或应急处理方式以及如何均衡”。现在的系统越来越复杂，如何合理的稳定这个确实很难。
前几天听到一个例子：公司上了Docker，然后Docker的数目是实际服务器和人员的数倍...
展开

作者回复: 谢谢你的分享，还举了一个很现实的例子。

8

lyonger
2020-04-06

亚马逊的复盘就提倡ask 5why方法，我觉得无论哪种方法，其关键就是故障处理，故障复盘，故障改进需做到整体的闭环，这就需要跳出故障本身，站在更上层考虑整体的业务。比如某个团队的a业务出了故障，找到了问题，自身业务已经修复，那么该问题其他团队的b/c/d业务也可能会遇到，那么如何推动线上的业务全部fix？ 我认为这需要一个良好的有效测试环境，毕竟对b/c/d...来讲是一次变更，要知道变更往往是故障的来源，所以变更...
展开

作者回复: 很棒的分享！

2

Jxin
2020-04-03

1.针对健壮性定责有点个人见解。首先，防御式编程没有毛病，但是防御式编程有时成本是比较大的。比如b服务依赖a服务，a服务做好代码健全成本很低，而b服务要做好完善的防御式编程成本较高，这时候b服务与a服务间的交互采用契约编程并不为过。
2.所以，我觉得在发生故障时，定责应该以a服务和b服务，谁做好代码健全的成本较低，谁负责主要责任。（a犯了很低级的错误，但能快速修复就无责。b服务因为a服务的失误...
展开

作者回复: 很好的分享。

针对最后提到的a犯了低级错误，这种其实我们定责时会要a承担对应的责任的，甚至是主要责任。

所以，具体case具体分析。

1
2

wholly
2020-04-07

重在改进，而不是处罚
展开

1

Geek_09c245
2020-04-06

老师，除了《SRE Google 运维解密》，还请推荐其他学习资料一二

作者回复: 我后面会统一推荐

2 1

Edward
2020-04-05

故障复盘其实跟研发项目复盘会议很相似，本质上都是围绕怎么去让“项目”做得更好。但是往往这种会议中，如果职位或地位比较高的人主导的观点或者方向错误时，底下的人员就会默默接受“挨骂”，比如提出前后端协作不顺畅问题导致项目延迟，最终根因变成单人积极性不足，这样打击了员工积极性的同事，也没有梳理痛点并改进，这样的氛围和项目接下来也是堪忧的。要形成这种“一起将事情做得更好的”文化氛围，个人觉得领...

展开

作者回复: 感同身受。你说地非常对，往大了说这个是文化问题，往细了说，团队中有影响力的人要能够发挥作用，“鼓励改进，而不是处罚错误”，这一点是要落到实处的。

感谢你的分享。

1

一步
2020-04-04

这里的应对故障的措施就是做好自己一切能做的，最大程度降低内部对外部的依赖

作者回复: 对，要遵循健壮性原则。

1

Helios
2020-04-04

经历过一次把case study变为批斗会的，研发、产品、质量的tl都在，全程质量部tl在骂人还是指名的骂人，还就骂一个人。

但是我就暗暗发誓，一定要听话，不再参加case study了。那时候是18年刚毕业时候参加的，后续看到开源社区的case study的文档，发现他们是为了项目的发展去复盘，而不...

展开

作者回复: 除了CL这个角色，更重要的是故障复盘的文化和机制，复盘文化往往也体现了公司的文化，是大家齐心协力共改进，还是出现问题一定要撇清责任，这一点不是技术问题，一定要从文化和氛围层面去解决。